Birzeit University - Faculty of Engineering and Technology
Electrical & Computer Engineering Department - ENCS4330
Real-Time Applications & Embedded Systems - 1st semester - 2024/2025

Project #1
Signals & Pipes under Linux
Due: November 9, 2024

Instructor: Dr. Hanna Bullata

Bungee jumping game simulation

We would like to create a multi-processing application that simulates the behavior of 3 teams of players playing bungee jumping from a bridge using signals and pipes facilities (see Figure 1). The simulated game can be explained as follows:

Figure 1: A bungee jumper in action.

1. We'll call the three teams **team_A**, **team_B** and **team_C** consecutively.

2. Each team is composed of 3 players numbered 1 to 3. Initially all players have a high level of energy. Of course that energy decreases with time and effort put while playing.

   In addition, a referee is needed as well to keep the score of each team and to coordinate actions as described below.

3. Upon a signal from the referee, the first member in each team will do a bungee jumping while the other 2 team members are watching on the bridge. Once the player who did the bungee jumping stabilizes after a random period of damped oscillations (e.g. almost still status), the 2 other team members need to pull him up to the bridge top. Pulling the bungee jumper takes a random period as well depending on the level of energy of the 2 pulling players.

   The referee is the one who decides if a bungee jumper has stabilized hanging down there so that the other 2 team members start pulling him.

   Keep in mind as well that the random periods spent to pull jumpers and to prepare for the next jumps are function of the energy level each player has. The energy level of players decreases with **(a)** each bungee jump and **(b)** every bungee jumper pull and **(c)** time.

4. When the bungee jumper has been pulled back to the bridge, the referee will compute a score that is inversely proportional to the time the bungee jumper spent in his jump to stabilize. In addition, the next team player will do the bungee jumping as quickly as possible to gain time (of course some random time will be wasted to prepare for the jump and once the referee allows the next jumps to take place once he's sure the jumper is well prepared).

5. The same sequence described above continues forever until any of the 3 teams has reached or exceeded a score provided by the user who runs the simulation.

6. The simulation ends if any of the following is true:

   - The user-defined amount of time allocated to the game is over.
   - One of the teams has got a score that exceeds a user-defined number provided by the user.

What you should do

- In order to implement the above-described application, you need to use the signals and pipes facilities. Be wise in the choices you make and be ready to convince me that you made the best choices :-)

- Write the code for the above-described application using a multi-processing approach.

- In order to avoid hard-coding values in your programs, think of creating a text file that contains all the game settings (e.g. values and ranges that should be user-defined) and give the file name as an argument to the main program. That will spare you from having to change your code permanently and re-compile.

- Use graphics elements from opengl library in order to best illustrate the application. Nothing fancy, just simple and elegant elements are enough.

- Test your program.

- Check that your program is bug-free. Use the `gdb` debugger in case you are having problems during writing the code (and most probably you will :-). In such a case, compile your code using the `-g` option of the `gcc`.

- The project lead is responsible to send the zipped folder that contains your source code and your executable before the deadline on behalf of the team. If the deadline is reached and you are still having problems with your code, just send it as is!